SETTLE: A Tuple Space Benchmarking and Testing Framework

Gregory M. Kapfhammer,
   Daniel M. Fiedler
   Kristen Walcott,
   Thomas Richardson
Department of Computer Science
Allegheny College

Ahmed Amer,
Panos Chrysanthis
Department of Computer Science
University of Pittsburgh
Contributions

- Benchmarking framework that can measure throughput and response time while varying the number of clients
- Tuple space aging technique that automatically populates tuple space before benchmark execution
- Detailed empirical study that evaluates tuple space performance, time overhead associated with aging, and impact of aging on space performance
Introduction to Tuple Spaces

Space clients can write, take, and read Entry objects
SETTLE Approach

- $q$ space clients execute the same benchmark in phases
- Client $C_j$ starts up $T_{delay} \in [T_{min}, T_{min} + V]$ ms after $C_{j-1}$
- Client pauses for $T_{delay}$ ms between the write and take phases
- Measure response time, $R(S_i, C_j, O)$, and throughput, $X(S_i, O, q)$
tuple space aging: preliminaries

- Could execute benchmark with an empty tuple space but \texttt{take} will execute faster than normal
- Age with either automatically generated or recorded/derived workloads
- \{r, t, w\}-frequency defines the fraction of the workload that will be associated with each space operation
- Define a frequency for each possible \texttt{Entry} type so that

\[
\text{freq(Null)} = 0.45 + \text{freq(String)} = 0.15 + \text{freq(Array)} = 0.25 + \cdots
\]

\[
\cdots + \text{freq(File)} = 0.15 = 1
\]
Tuple Space Aging: Example

- Automatically populate space with Entry objects of same type but different field values.
Experiment Design

- Dual Intel Xeon Pentium III processors and 512 MB main memory
- GNU/Linux kernel 2.4.18-14smp, Java 1.4.2 compiler, Java 1.4.2 VM in HotSpot client mode, Jini 1.2.1
- LinuxThreads 0.10 was configured with a one-to-one mapping between Java threads and kernel processes
- Clients $C_1, \ldots, C_q$ executed on the same machine as JavaSpace $S_i$
- Other configurations are possible and additional experiments are currently being conducted
## Experiment Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{min}}$</td>
<td>200 ms</td>
</tr>
<tr>
<td>$V$</td>
<td>50 ms</td>
</tr>
<tr>
<td>$T_{\text{delay}}$</td>
<td>[200, 250] ms</td>
</tr>
<tr>
<td>Number of Entry Objects</td>
<td>{1000}</td>
</tr>
<tr>
<td>Aging Workload Size ($</td>
<td>W</td>
</tr>
<tr>
<td># of Clients (non-aged) ($q$)</td>
<td>{2, 8, 14, 22}</td>
</tr>
<tr>
<td># of Clients (aged) ($q$)</td>
<td>{8, 14}</td>
</tr>
<tr>
<td>{$r, t, w$}-frequency</td>
<td>{0, 0, 100}</td>
</tr>
<tr>
<td>Entry Objects</td>
<td>{Null, String, Array, File}</td>
</tr>
</tbody>
</table>
When space is not aged, throughput knees at 8 or 14 clients.
NullIO: Response Time, Throughput

When throughput knees at 8 clients, average response time continues to increase linearly.
When throughput knees at 14 clients, average response time continues to increase linearly.
When $|W| = 3000$ there is a 30% increase in NullIO execution time.
NullIO: Aging Time Overhead

Aging never consumes more than 31% of entire benchmark time
Cleaning incurs less time overhead than aging due to snapshot
Aging’s Impact on Throughput

Aging reduces tuple space throughput as workload size increases.
Aging increases tuple space response time as workload size increases.
Related Work

- Bulej et al. focus on regression benchmarking
- Sterk et al. evaluate tuple space performance in the context of bioinformatics
- Noble and Zlateva measure tuple space performance for astrophysics computations
- Hancke et al. and Neve et al. measure tuple space performance through the use of statistically guided experiments
- Smith and Seltzer introduced file system aging
Future Work

- Additional experiments: (i) transient vs. persistent tuple spaces, (ii) remote client interactions, (iii) different tuple space implementations, (iv) new versions of Jini and JavaSpaces
- Workload studies for tuple space-based applications
- Additional micro, macro, and application-specific benchmarks
- Definition-use testing for tuple space-based applications: how do you know your application puts the right data into the space?
Conclusions

- SETTLE measures throughput and response time and supports automatic tuple space aging
- In current SETTLE configuration, JavaSpaces can support between eight and fourteen concurrent local clients without reducing average response time
- Tuple space aging can be performed with acceptable time overhead
- Aging does support the characterization of worst-case performance
Resources


  [http://cs.allegheny.edu/~gkapfham/research/settle/](http://cs.allegheny.edu/~gkapfham/research/settle/)